ELSEVIER Linear Algebra and its Applications 275–276 (1998) 281–286

LINEAR ALGEBRA AND ITS APPLICATIONS

The Gauss–Huard algorithm and LU factorization

Walter Hoffmann

Faculty of Mathematics, Computer Science, Physics and Astronomy University of Amsterdam, Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands

Received 29 October 1996; accepted 28 April 1997

Submitted by V. Mehrmann

Abstract

In this paper we analyze the Gauss–Huard algorithm. From a description of the algorithm in terms of matrix–vector operations we reveal a close relation between the Gauss–Huard algorithm and an LU factorization as constructed in an *ikj* variant. © 1998 Elsevier Science Inc. All rights reserved.

Keywords: Gauss–Huard algorithm; LU factorization

1. Introduction

A 'new method' for solving dense linear systems was published in 1979 [10]. This method was recognized as an efficient variant of the Gauss–Jordan algorithm and has since become known as the Gauss–Huard algorithm [1]. (A theoretical relation with 'the conjugate direction algorithm' was demonstrated by Laurent-Gengoux and Trystram [11].) A drawback of the Gauss–Huard algorithm seemed to be the fact that it appeared not to allow for a stabilizing pivoting strategy and therefore was not applicable for general matrices.

After demonstrating that in Gauss–Jordan elimination the application of *row pivoting* (based on *column* interchanges) yields a numerically stable implementation [2] (as opposed to the nonstable version that uses partial pivoting by *column pivoting*, see [12]), Dekker et al. [3] showed the applicability of the row pivoting strategy in Gauss–Huard elimination. The numerical stability of the Gauss–Huard algorithm with row pivoting is proven in [4].

PII: S0024-3795(97)10021-0

In the current paper we describe a relation of the Gauss–Huard algorithm with LU factorization, or more precisely, with one out of essentially six variants of LU factorization [5].

2. The original Gauss–Huard algorithm

As we want to demonstrate a theoretical relation between the Gauss–Huard algorithm and LU factorization, we will not consider the subject of pivoting to achieve a numerically stable algorithm; for a version of the Gauss–Huard algorithm that includes partial pivoting we refer to [3,9]. For our comparison we start with a description of the Gauss–Huard algorithm in its original formulation [1,10]. (see Fig. 1).

The algorithm transforms a given nonsingular matrix into an identity matrix by linear operations on rows of the matrix.

Assume that after $(i-1)$ steps an identity matrix of order $(i-1)$ has been created in the upper left-hand corner of the matrix. We then demonstrate that in the ith step of the algorithm an identity matrix of order i will be constructed, thus demonstrating the working of the algorithm.

In lines 2–6, the first $(i-1)$ elements of row i are annihilated by subtracting appropriate multiples of earlier rows; this is called the 'row-elimination' part.

In line 7 the ith row is scaled such that the value 1 appears on the diagonal; this is called the 'scaling' part.

In lines 8–12, the first $(i-1)$ elements of column i are annihilated by subtracting appropriate multiples of row i; this is called the 'column-elimination' part.

At this point, an $i \times i$ identity matrix has been created in the upper left-hand corner and this describes the working of the algorithm.

```
1  for i = 1 to n do
2    for k = 1 to i - 1 do
3      for j = i to n do
4        α_ij := α_ij − α_ik × α_kj
5      enddo
6    enddo
7    for j = i + 1 to n do α_ij := α_ij/α_ii
8    for k = 1 to i - 1 do
9      for j = i + 1 to n do
10       α_kj := α_kj − α_ki × α_ij
11     enddo
12   enddo
13 enddo
```

Fig. 1. Gauss–Haurd elimination without pivoting.

A linear system is solved by treating the right-hand side as if it were the $(n+1)$th column of the matrix. After n steps, the matrix will be transformed into the identity matrix and the solution will be found in the $(n+1)$th column.

The number of floating-point operations needed in the Gauss–Huard algorithm turns out to be equal to the number of floating-point operations in an LU factorization: $\frac{2}{3}n^3 - \frac{1}{2}n^2 + \frac{5}{6}n$.

3. A relation with LU factorization

For a comparison of the Gauss–Huard algorithm with LU factorization, we focus on the existence of six variants in the data flow for computing an LU factorization as has been introduced in [5]. These six variants can be discriminated pairwise by one out of three fundamental operations which are at the basis of that specific variant:

1. Dot product of two vectors for the *ijk* and *jik* variant.
2. Matrix–vector multiplication for the *ikj* and the *jki* variant.
3. Rank-one matrix update for the *kij* and the *kji* variant.

For detailed use of these fundamental operations in the context of LU factorization we refer to [6], [7] or [8]. In the sequel we will demonstrate that the Gauss–Huard algorithm is closely related to an *ikj* form of LU factorization; in particular the variant with a standardization of diagonals such that U has unit diagonal elements. With this choice the factorization scheme is described in Fig. 2

For a given value of i, lines 2–6 define the calculation of the ith rows of L and U. We will describe this calculation in a mathematical way as an alternative to the algorithmic description.

Let i be a given integer, $1 < i \leqslant n$, and assume that the first $i-1$ rows of L and U have already been calculated. Our goal is to define the calculation of the ith row of L and the ith row of U using matrix terminology.

The first i rows of L and U in partitioned form look like

```
1  for i = 1 to n do
2    for k = 1 to i - 1 do
3      for j = k + 1 to n do
4        α_ij := α_ij - α_ik × α_kj
5      enddo
6    enddo
7    for j = i + 1 to n do α_ij := α_ij/α_ii
8  enddo
```

Fig. 2. *ikj* factorization without pivoting (U unit diagonal).

$$\begin{pmatrix} L_{i-1} & \underline{0} & O \\ m^{\mathrm{T}} & \mu & \underline{0}^{\mathrm{T}} \end{pmatrix} \text{ and } \begin{pmatrix} U_{i-1} & v_{i-1} & R_{i-1} \\ \underline{0}^{\mathrm{T}} & 1 & w^{\mathrm{T}} \end{pmatrix}, \text{respectively,}$$

where the following notation has been used:
L_{i-1}, U_{i-1} are the $(i-1) \times (i-1)$ leading submatrices of L and U, respectively,
$(m^{\mathrm{T}}\ \mu)$ is the ith row of L (which is to be calculated; μ being the diagonal element),
v_{i-1} the vector consisting of the first $(i-1)$ elements of the ith column of U,
R_{i-1} the $(i-1) \times (n-i)$ sub matrix in the upper right-hand corner of U,
$(1\ w^{\mathrm{T}})$ the ith row of U (the row that is to be calculated, having 1 at the diagonal).

The calculations determine L_i and U_i holding

$$L_i = \begin{pmatrix} L_{i-1} & \underline{0} \\ m^{\mathrm{T}} & \mu \end{pmatrix} \text{ and } U_i = \begin{pmatrix} U_{i-1} & v_{i-1} \\ \underline{0}^{\mathrm{T}} & 1 \end{pmatrix}.$$

Comparing parts of the ith row of A with the corresponding parts of the ith row in the product $L \times U$ yields the following relations

$$m^{\mathrm{T}} U_{i-1} = A[i, 1\!:\!i-1], \tag{1}$$

with the obvious notation of $A[k, l\!:\!m]$ for the elements l up to m of the kth row of matrix A. The above equation defines m as the solution of a triangular linear system with lower triangular matrix U_{i-1}^{T} and right-hand side $A[i, 1\!:\!i-1]^{\mathrm{T}}$.

After m has been determined, the unknowns μ and w follow from the identity

$$(m^{\mathrm{T}}\ \mu) \times \begin{pmatrix} v_{i-1} & R_{i-1} \\ 1 & w^{\mathrm{T}} \end{pmatrix} = A[i, i\!:\!n], \tag{2}$$

yielding

$$\mu(1\ w^{\mathrm{T}}) = A[i, i\!:\!n] - m^{\mathrm{T}}(v_{i-1}\ R_{i-1}). \tag{3}$$

Relations 1 and 3 define the factorization as described in Fig. 2.

Now we come to our essential observations. The computation of vector m as defined in relation 1 need not be performed explicitly. Instead of using the computed value of m in formula (3), we use the algebraic expression for m that comes from formula (1):

$$m^{\mathrm{T}} = A[i, 1\!:\!i-1] U_{i-1}^{-1}. \tag{4}$$

This expression substituted in relation 3 yields:

$$\mu(1\ w^{\mathrm{T}}) = A[i, i\!:\!n] - A[i, 1\!:\!i-1](U_{i-1}^{-1} v_{i-1}\ U_{i-1}^{-1} R_{i-1}). \tag{5}$$

We observe that row-vector m (and in general, any row of L) need not be used explicitly, provided the quantities $U_{i-1}^{-1} v_{i-1}$ and $U_{i-1}^{-1} R_{i-1}$ are available.

Now assume that instead of vector v_{i-1} we could deal with $U_{i-1}^{-1}v_{i-1}$ and likewise, assume that instead of R_{i-1} we could deal with $U_{i-1}^{-1}R_{i-1}$. Then the issue is to find (preferably simple) update rules for creating $U_i^{-1}v_i$ .and $U_i^{-1}R_i$ (or in combined form, an update rule for $U_i^{-1}(v_i\ R_i)$).

First we formulate an expression for U_i^{-1} with $U_i = \begin{pmatrix} U_{i-1} & v_{i-1} \\ \underline{0}^{\mathrm{T}} & 1 \end{pmatrix}$.

It is easy to verify that this is given by $U_i^{-1} = \begin{pmatrix} U_{i-1}^{-1} & -U_{i-1}^{-1}v_{i-1} \\ \underline{0}^{\mathrm{T}} & 1 \end{pmatrix}$.

Next we observe: $(v_i\ R_i) = \begin{pmatrix} R_{i-1} \\ w^{\mathrm{T}} \end{pmatrix}$ from which we arrive at

$$U_i^{-1}(v_i\ R_i) = \begin{pmatrix} U_{i-1}^{-1} & U_{i-1}^{-1}v_{i-1} \\ \underline{0}^{\mathrm{T}} & 1 \end{pmatrix}\begin{pmatrix} R_{i-1} \\ w^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} U_{i-1}^{-1}R_{i-1} - U_{i-1}^{-1}v_{i-1}w^{\mathrm{T}} \\ w^{\mathrm{T}} \end{pmatrix}. \quad (6)$$

Recall our assumption that we can deal with the matrix $U_{i-1}^{-1}R_{i-1}$ and the vector $U_{i-1}^{-1}v_{i-1}$, then formula (6) is just the required update formula for $U_i^{-1}(v_i\ R_i)$ which appears to be a rank-one update of matrix $U_{i-1}^{-1}R_{i-1}$ with the matrix product $-(U_{i-1}^{-1}v_{i-1})w^{\mathrm{T}}$ combined with adding w^{T} as the next row.

A comparison with the original Gauss–Huard algorithm as presented in Fig. 1 shows that exactly this rank-one update is performed in lines 8–12; the part of the matrix to be updated, $(U_{i-1}^{-1}R_{i-1})$, being present in the upper right-hand corner of the matrix; the column-vector $(U_{i-1}^{-1}v_{i-1})$ being available in the first part of the ith column and the row-vector w^{T} being available in the last part of the ith row.

As a conclusion we observe that the Gauss–Huard algorithm calculates in each step the next row of a unit-diagonal matrix U that satisfies $A = LU$ where L is a lower triangular, like is being done in the ikj form of LU factorization; compare lines 2–6 in Fig. 1 with lines 2–6 in Fig. 2. We should remark however, that in the Fig. 2 algorithm also the next row of L is calculated. This corresponds with a different range of the index j.

In the Gauss–Huard algorithm, the row of U that has been calculated will not be kept, as the the upper triangular part of the given matrix will be overwritten by the inverse of matrix U, this being the result of repeatedly applying the rank-one updates in the algorithm.

As a final remark, it should be noted that the lower triangular part of the matrix, its diagonal included, remains unchanged in the Gauss–Huard algorithm.

References

[1] M. Cosnard, Y. Robert, D. Trystram, Résolution parallèle de systèmes linéaires denses par diagonalisation, E.D.F. Bulletin de la Direction des Etudes et des Recherches C(2) (1986) 67–87.

[2] T.J. Dekker, W. Hoffmann, Rehabilitation of the Gauss-Jordan algorithm, Numerische Mathematik 54 (1989) 591–599.
[3] T.J. Dekker, W. Hoffmann, K. Potma, Parallel algorithms for solving large linear systems, Journal of Computational and Applied Mathematics 50 (1994) 221–232.
[4] T.J. Dekker, W. Hoffmann, K. Potma, Stability of the Gauss-Huard algorithm with partial pivoting Computing 58 (1997) 225–244.
[5] J.J. Dongarra, F.G. Gustavson, A. Karp, Implementing linear algebra algorithms for dense matrices on a vector pipeline machine, SIAM Review 26 (1984) 91–112.
[6] T.L. Freeman, C. Phillips, Parallel Numerical Algorithms, Prentice-Hall International Series in Computer Science, Prentice-Hall, Englewood Cliffs, NJ, 1992.
[7] G.H. Golub, C.F. Van Loan, Matrix Computations, 2nd edn., Johns Hopkins Univ. Press, Baltimore, MD, 1989.
[8] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.
[9] W. Hoffmann, K. Potma, G. Pronk, Solving dense linear systems by Gauss-Huard's method on a distributed memory system, Future Generation Computer Systems 10 (1994) 321–325.
[10] P. Huard, La méthode du simplexe sans inverse explicite, E.D.F. Bulletin de la Direction des Etudes et des Recherches Série C 2, 2 (1979).
[11] P. Laurent-Gengoux, D. Trystram, A new presentation of the conjugate direction algorithm, Journal of Computational and Applied Mathematics 32 (1990) 417–422.
[12] G. Peters, J.H. Wilkinson, On the stability of Gauss-Jordan elimination with pivoting, Communications of the ACM 18 (1975) 20–24.